**PROGETTO OOP a.a 2017-2018**

Gruppo Progetto : Di Egidio Laura 248831

Gentile Fabio 248809

Funzionalità del sistema:

Il nostro sistema deve gestire ed organizzare un insieme di collezioni, servizi e persone. L’ obiettivo del nostro sistema è la consultazione di manoscritti che devono essere digitalizzati. Le opere devono poter essere pubblicate e successivamente visionate da tutti gli utenti iscritti.

**1.1 Analisi dei requisiti**

Successivamente i requisiti per ogni sottosistema che siamo riusciti a trovare sono i seguenti:

**1 VIEWER**

* 1. **Login/registrazione** : Funzione per accedere al sistema da parte di utenti registrati. Possibilità di registrazione per nuovi utenti .
  2. **Ricerca** : Un utente può svolgere la ricerca nel catalogo per metadati (anno, titolo..) oppure all’ interno del testo della trascrizione. Funzione di ricerca dei testi attraverso criteri di scelta.
  3. **Download dell’ opera** : Operazione a disposizione solo per utenti premium (immaginiamo ad esempio che l’ utente premium è per noi un utente abbonato). Funzionalità che permette agli utenti premium di poter scaricare l’ opera.
  4. **ModuloTrascrittore** : Un utente può diventare un collaboratore e trascrivere testi compilando un modulo. Funzionalità per tutti gli utenti registrati di diventare trascrittori delle opere.
  5. **Dati Utente** : Funzionalità da parte dell’utente di poter accedere al proprio profilo e visualizzare i dati personali.

**2 UPLOADER**

**2.1 Gestione immagine** : In questo package distinguiamo il **Revisore Immagine** con le seguenti funzionalità:

* **Elimina immagine:** Eliminazione delle immagini non conformi alla qualità standard
* **Convalida immagine:** Funzione di convalida dell’ immagine scansionata dall’ Acquisitore.
* **Revisiona immagine:** Funzione di controllo dell’ immagine scansionata.

Distinguiamo ora l’ **Acquisitore** :

* **Scansione immagine** : Funzionalità che permette l’ acquisizione di immagini ad alta risoluzione dei manoscritti. L’ acquisitore caricherà le immagini in formato ‘jpg’ assegnandole all’opera di riferimento.

1. **TRANSCRIBER**
   1. **Gestione Testo** : In questo package distinguiamo il **Revisore Testo** e il **Transcriber:**

**Revisore Testo** :

* **Revisiona Testo :** Funzione da parte del Revisore di controllo del testo in formato TEI
* **Convalida Testo :** Funzione che permette al Revisore di convalidare il Testo trascritto
* **Modifica Testo :** Funzionalità che permette la modifica del testo da parte del revisore.

**Transcriber :**

* **Trascrive Testo :** Funzione di trascrizione testo da parte di trascrittori

**4 MANAGER**

4.1 **Assegnazione opere** : Tale funzionalità gestisce le assegnazioni di una o più opere ad uno o più trascrittori.

4.2 **Revisiona Trascrizioni** : Funzione di revisione delle trascrizioni da parte del Manager

4.3 **Pubblicazione :** Funzionalità di pubblicazione delle trascrizioni e delle opere.(IMMAGINI)

4.4 **Correzione Trascrizione** : Funzione di modifica delle trascrizioni da parte del Manager

4.5 **Convalida Testo** : Funzione di convalida delle trascrizioni da parte del Manager

4.6 **Assegna feedback :** (1-5) Funzione di gestione del livello dei trascrittori. Il livello più alto è per noi il punteggio 5, di base tutti i trascrittori partiranno con un livello standard di 3.

* 1. **Revisione immagine digitale** : Funzionalità di controllo dell’ acquisizione delle immagini

1. **ADMINISTRATOR**

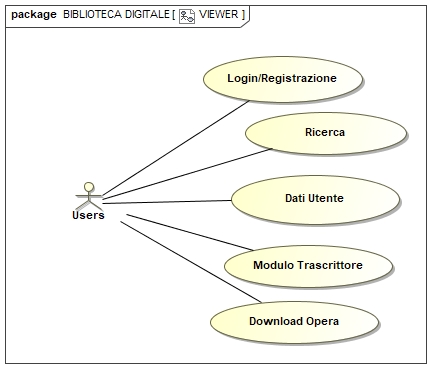
5.1 **Cancellazione Utente** : Funzionalità che permette di cancellare utenti dal sistema.

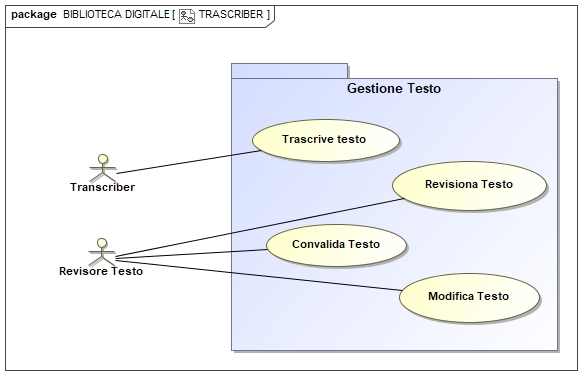
5.2 **Promozione Utente** : Funzione che permette di promuovere un Utente da Utente base a Trascittore oppure a Utente Premium

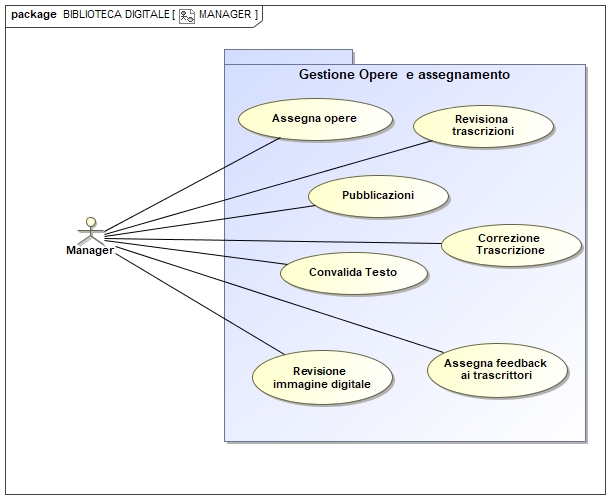
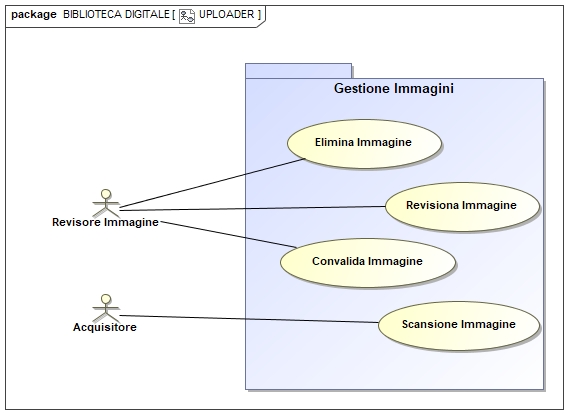
5.3 **Inserimento Utente** : Funzione che permette l’ inserimento di utenti all’ interno del sistema.

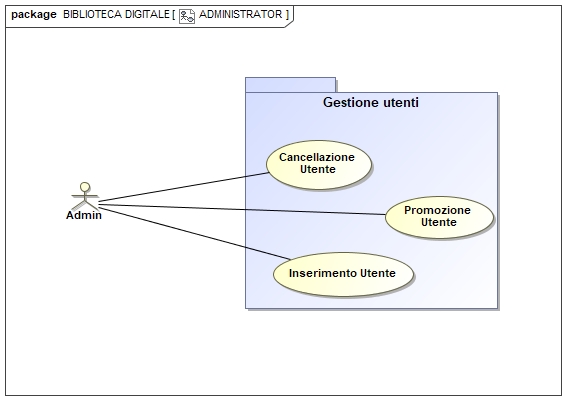
1.2 MODELLI UML USE CASE

Per definire al meglio lo Use Cases Diagram completo abbiamo di nuovo suddiviso gli attori con le loro funzionalità nei sottosistemi

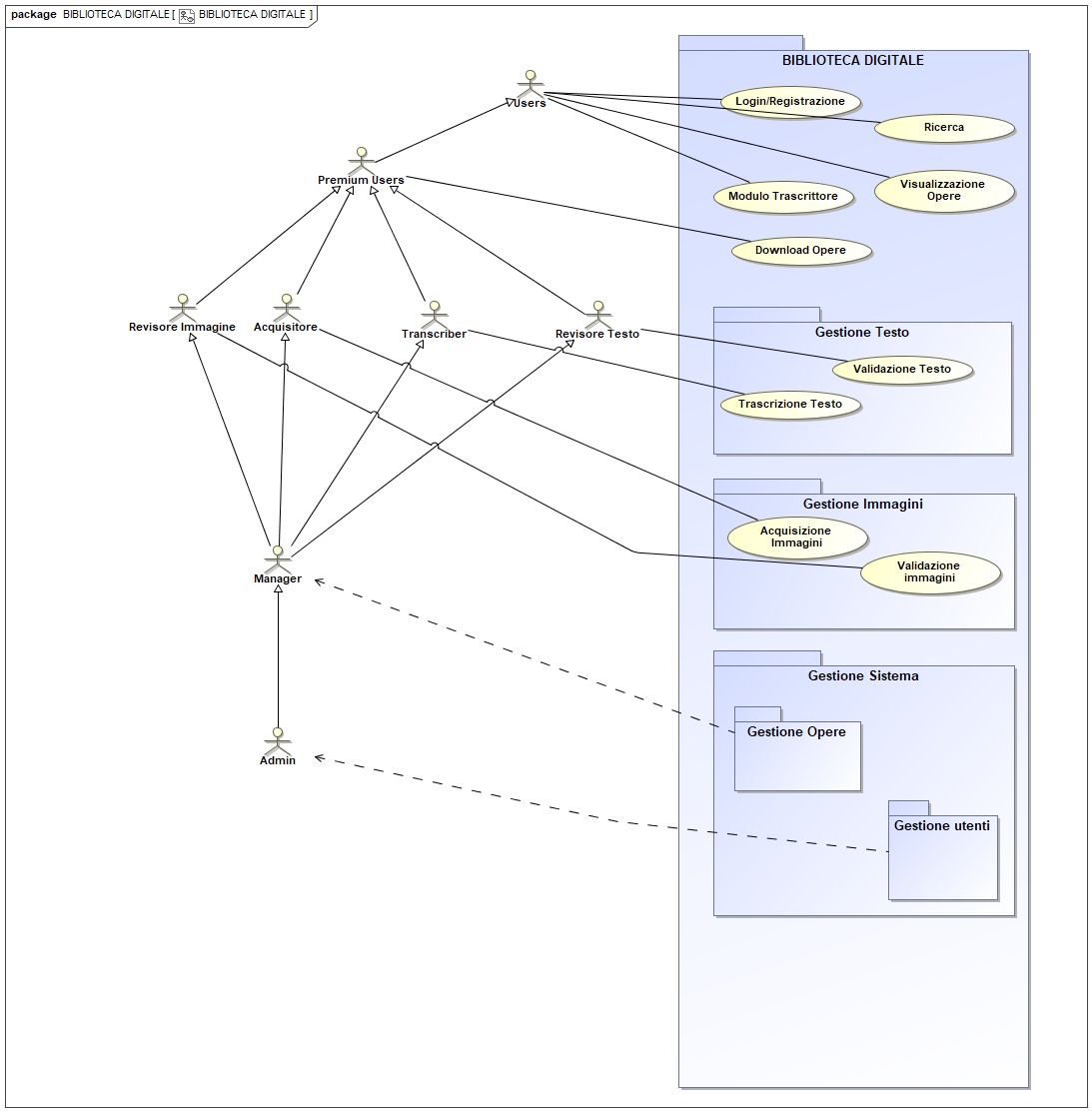




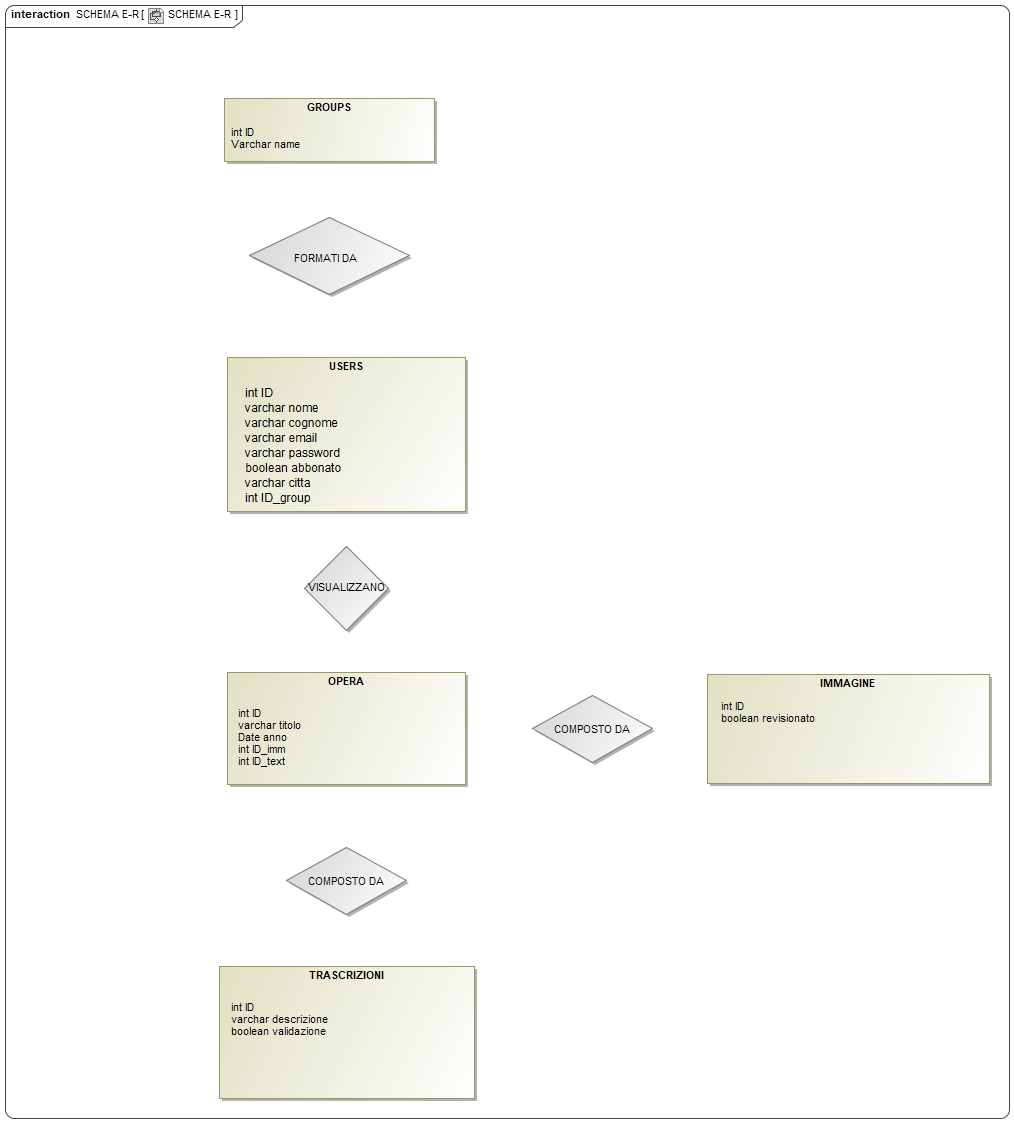




UML USE CASES COMPLETO :



Prima di determinare il Modello di Dominio è stato utile, per una maggior visione delle entity, crearci lo schema ER.

****

(1,n)

(1,n)

(1,1)

(1,1)

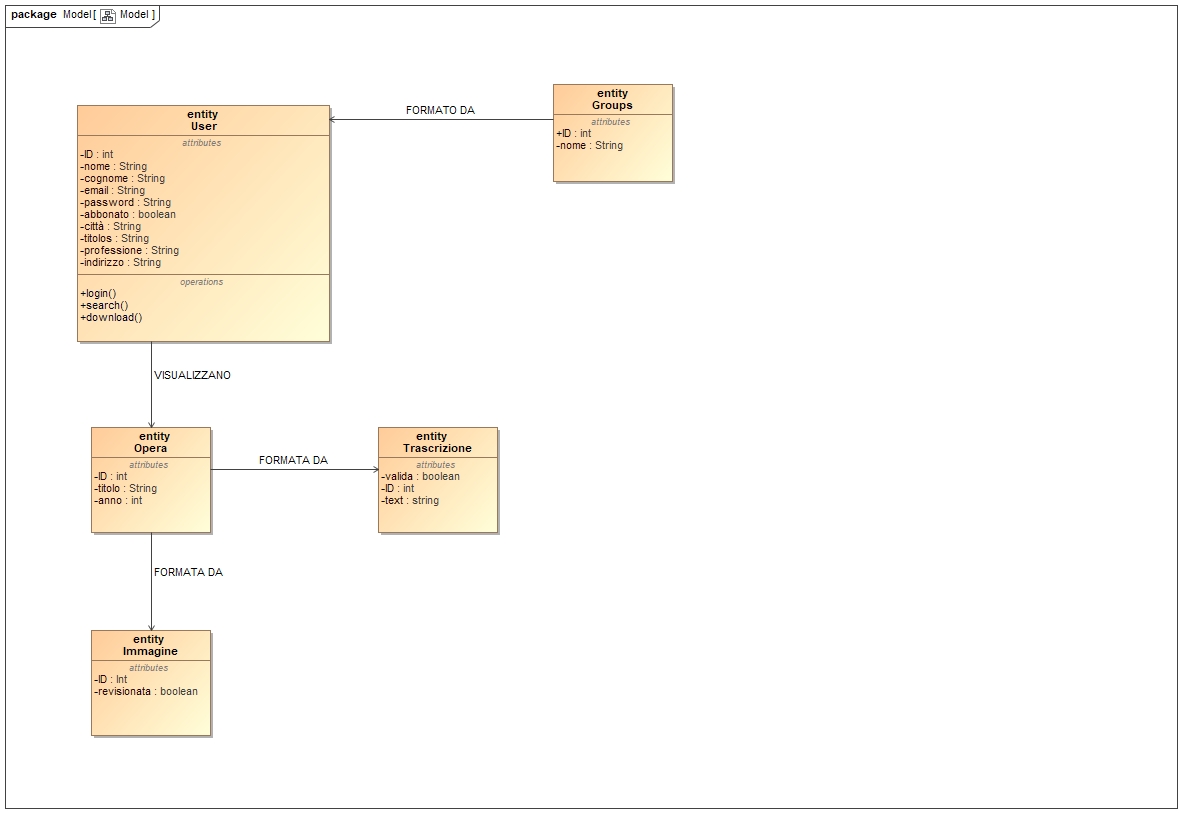
(1,n)

(1,n)

(1,1)

(1,n)

1.3 DOMAIN MODEL



1.4 ANALISI FINALIZZATA ALL’INDIVIDUAZIONE DELLE CLASSI ENTITY BOUNDARY E CONTROLLER

**ENTITY**

|  |  |  |
| --- | --- | --- |
| ENTITY | ATTRIBUTI | DEFINIZIONE |
| **Groups** | ID  Nome | In base all’ ID e al nome del tipo ‘manager, trascrittore...’ capiamo quale ruolo svolgo il nostro utente all’interno del sistema. Tutti gli utenti sono per noi utenti registrati ma che in base al ruolo definito appunto da Groups siamo in grado di attribuirgli determinate azioni. |
| **Users** | ID  Nome  Cognome  Abbonato  Email  Password  Citta  Indirizzo  Titolo di studio | Nella classe Users l’utente può visualizzare il proprio profilo determinato dai seguenti attributi. Inoltre, l’utente può svolgere le azioni quali: login,  registrazione, modifica,  revisione, pubblicazione,  download, trascrizione,  acquisizione, feedback… |
| **Opera** | ID  Titolo  Anno | Opera formata da un codice ID, il titolo dell’opera e l’anno di pubblicazione. |
| **Immagine** | ID  Revisionato | Nell’immagine abbiamo l’attributo revisionato (boolean) da dove specifichiamo se la scansione di una certa opera è stata revisionata successivamente |
| **Trascrizione** | ID  Convalida  Text | Anche la trascrizione ha un attributo booleano ‘convalida’ che specifica se la trascrizione sia stata o meno convalidata. |

**BOUNDARY**

|  |  |
| --- | --- |
| **Loginform** | Classe che permette l’accesso al sistema e la visualizzazione dei dati utente che conterrà tutti i metodi relativi all’accesso al sistema e alla visualizzazione |
| **Browseform** | Classe che permette di visualizzare le opere, cercare attraverso dei criteri ed eventualmente effettuare il download |
| **DiventaTrascrform** | Classe che permette di compilare il modulo per diventare trascrittore |
| **Uploadform** | Classe che permette di caricare le immagini scansionate |
| **Trascrform** | Classe che permette la trascrizione e l’invio delle trascrizioni |
| **GestioneOpform** | Classe che permette di modificare/eliminare/revisionare/convalidare le opere |
| **GestioneUtform** | Classe che permette la gestione degli utenti, in particolare di eliminare/promuovere/inserire gli utenti |
| **Feedbackform** | Classe che permette di gestire i feedback di determinati utenti, in particolare assegnare/eliminare feeback |
| **Pubblicaform** | Classe che permette la pubblicazione delle opere |
| **AssegnaOpform** | Classe che permette l’assegnamento delle opere a determinati utenti |

**CONTROL OBJECT**

Analizzando le funzionalità del sistema abbiamo individuato diversi controller che attraverso i loro metodi consentono la corretta interazione col database

**LoginController()** : Classe che prende in input i dati User/Password e verifica se l’utente è registrato e che i dati siano corretti

**BrowseController()** : Classe che prende in input i criteri di ricerca, verifica che sia presente nel DB e restituisce l’opera richiesta in termini di visualizzazione oppure di Download

**DiventaTrascrController()** :Classe che prende in input il modulo compilato e lo inoltra all’admin per l’accettazione  
**UploadController()**: Classe che prende in input l’ immagine scansionata in .JPG e la inoltra nel sistema  
**TrascrController()**:Classe che prende in input la trascrizione in formato .txt e la inoltra nel sistema

**GestioneOpController()** : Classe che prende in input un comando (modifica/elimina/revisiona/convalida e un opera selezionata) e permette la modifica dei dati  
**GestioneUtController()**:Classe che prende in input un comando (elimina/promuove/inserire ed un utente) e permette la modifica dei dati

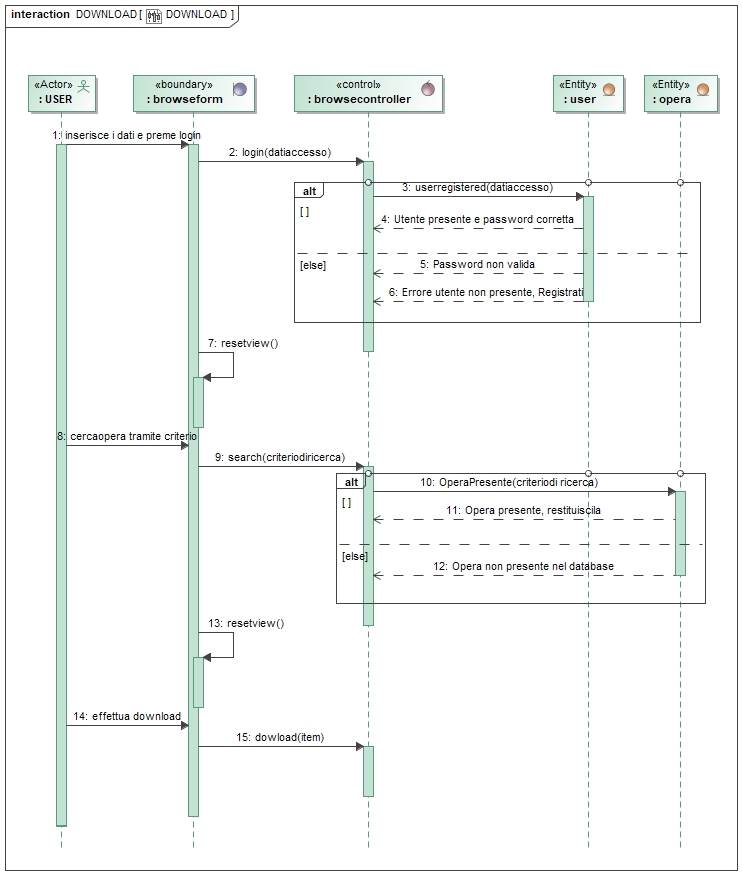
**FeedbackController()**:Classe che prende in input un comando(Assegna/Rimuovi ed un utente trascrittore) e permette la modifica del feedback assegnato allo stesso

**PubblicaController()**:Classe che prende in input un comando e permette la modifica dello stato dell’opera oppure della semplice scansione da privata a pubblica

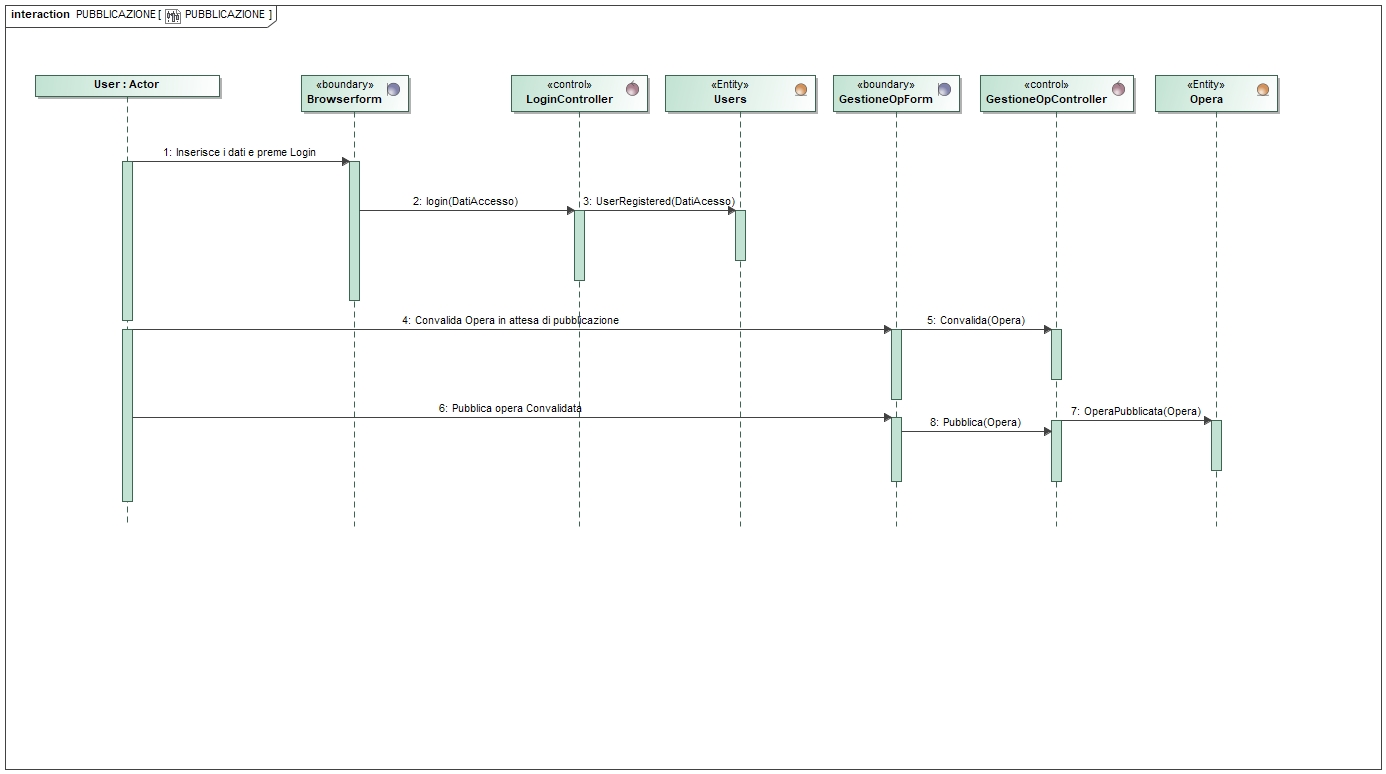
**AssegnaOpController()**:Classe che prende in input un comando e permette l’assegnamento di un opera ad un determinato utente

Esempi di iterazione con il sistema:

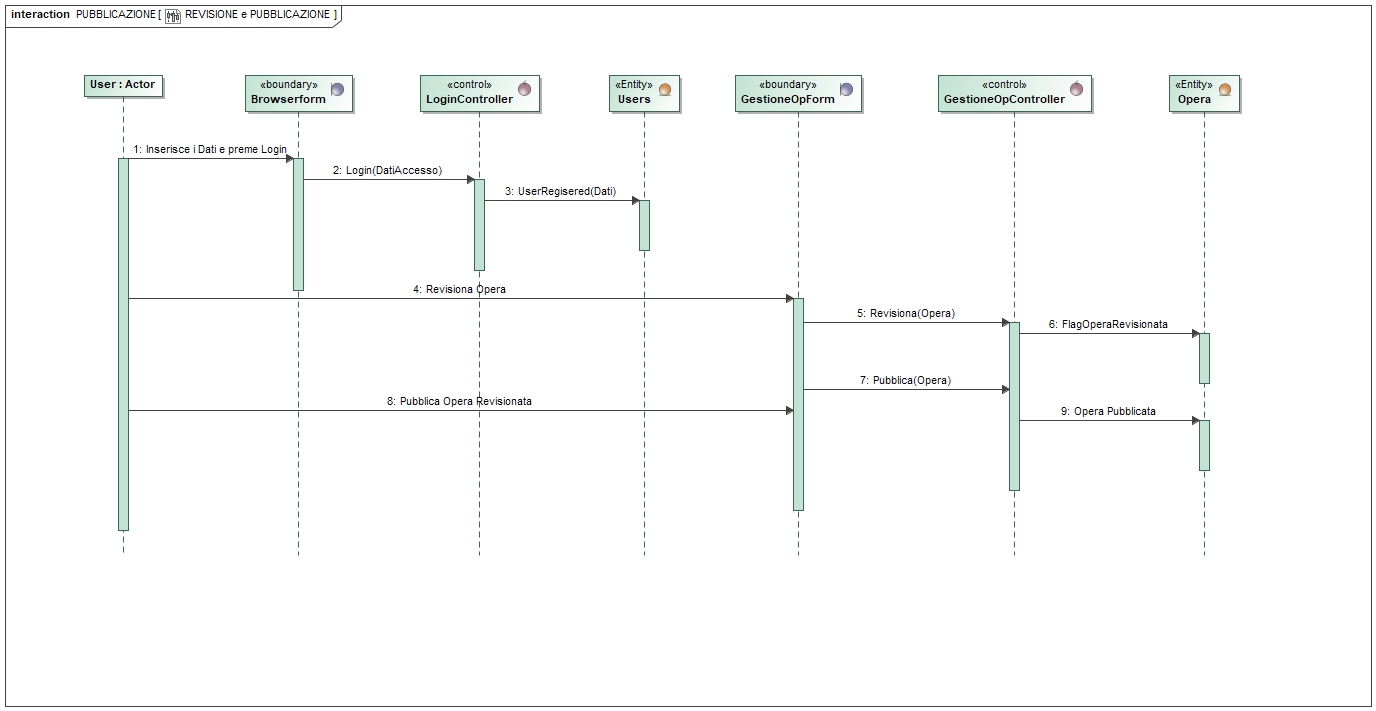
DOWNLOAD OPERA



PUBBLICAZIONE OPERA

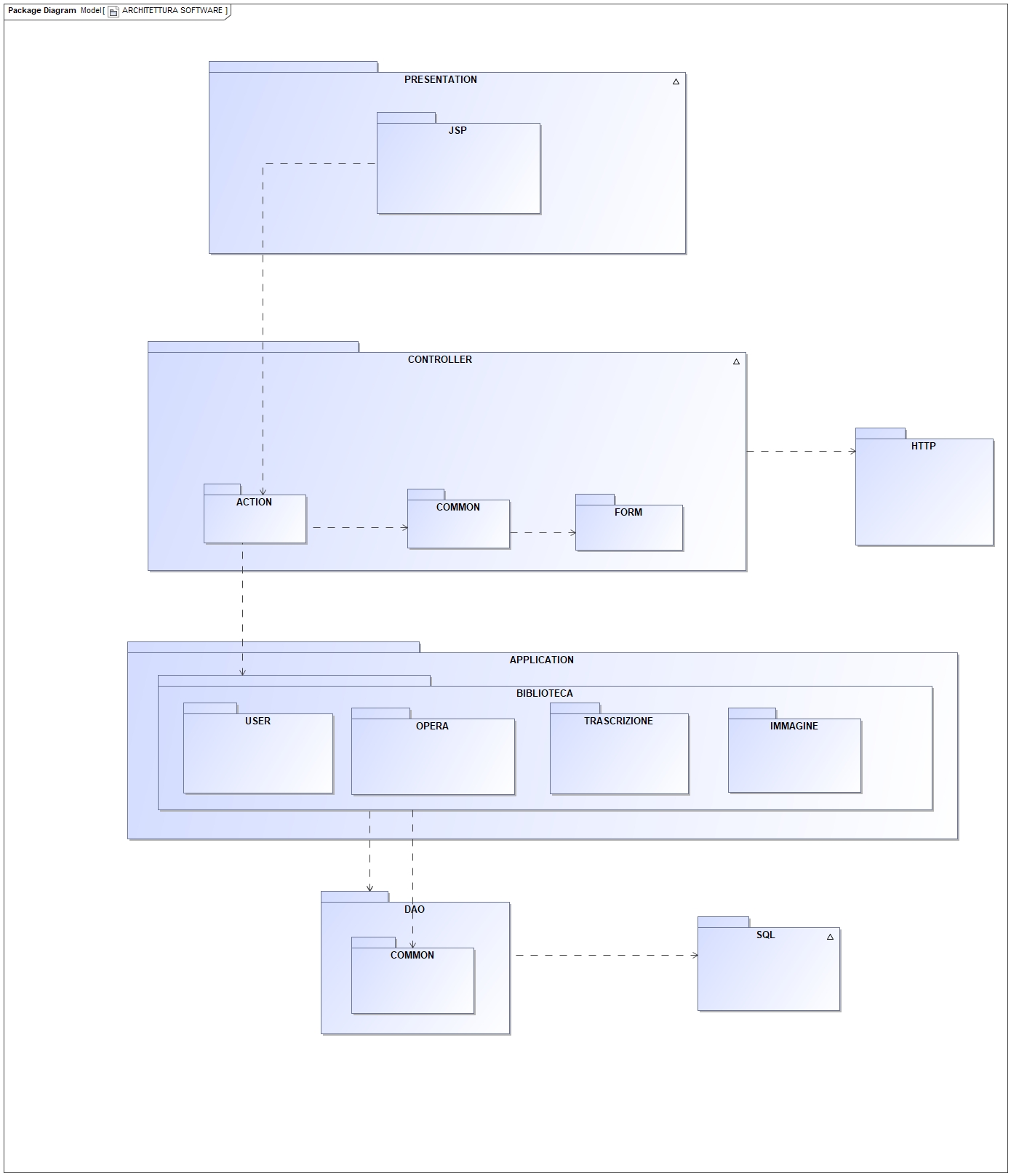
****

REVISIONE E PUBBLICAZIONE



2 SYSTEM DESIGN

2.1 ARCHITETTURA SOFTWARE



2.2 DESCRIZIONE ARCHITETTURA

L’ architettura software descrive le tecniche da affrontare, tra le altre cose, persistenza, sicurezza, gestione degli errori e interfacce di sistema. I meccanismi architettonici devono essere modellati per dare agli sviluppatori una vista sui meccanismi comuni che trascendono ogni caso d’ uso. L'applicazione deve avere oggetti memorizzati in modo persistente; quindi, un database strato, chiamato il pacchetto DAO, deve essere aggiunto per fornire questo servizio.  
 La soluzione per una complessa applicazione è quella di utilizzare un database commerciale; tipicamente questo database è un RDBMS (Relation DataBase Management System) come Oracle o SQL. Inoltre, dato che il nostro sistema Biblioteca è destinato ad essere portatile, abbiamo scelto di utilizzare uno strumento di database open source MySQL. I dati sono manipolati usando DAO, implementando l’ interfaccia comune per chiamare operazioni comuni come ricerca(), download(), modifica() e pubblica() sugli oggetti.

2.3 PATTERN DESIGN

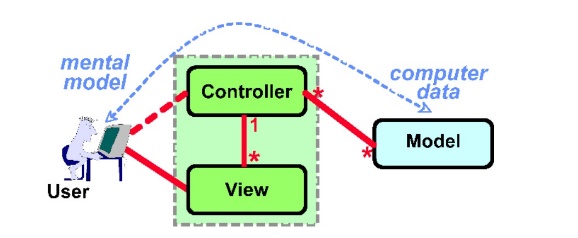
Sono stati scelti dal nostro gruppo diversi modelli per la progettazione del nostro sistema:

1. il pattern MVC (Model View Controller), in quanto permette una

suddivisione del sistema in parti indipendenti tra di loro più facili da gestire.

Il pattern architetturale è così diviso

* Rappresentazione del modello di dominio (Model)
* Interfaccia Utente (View)
* Controllo operazioni Utente-Sistema (Controller)



MVC è un pattern architetturale che ci permette di separare la logica di presentazione dei dati da quella di business.

I componenti software sono divisi in base ai compiti svolti all’interno del sistema

* Il Model fornisce i metodi per accedere ai dati utili. Gestisce direttamente i dati, la logica e le regole dell’applicazione.
* Il View si occupa della visualizzazione dei contenuti. È una qualsiasi rappresentazioni in output di informazioni, nel nostro esempio le immagini, il testo ecc..
* Il Controller riceve i comandi dall’ utente attraverso il view e li svolge alterando lo stato degli altri due componenti.

1. Il Controller: Questo modello di progettazione è fornito dal framework Struts

che supporta il pattern MVC. Fornisce un punto di ingresso centralizzato per la

gestione delle richieste.

1. Data Access Object (DAO) Pattern architetturale per la gestione della persistenza:
   * Classi con relativi metodi rappresentano entità tabellare di

un RDBMS

* Usato principalmente in applicazioni web di tipo Java EE,

per stratificare e isolare l'accesso ad una tabella tramite

query (poste all'interno dei metodi della classe)

* I metodi del DAO con le rispettive query dentro verranno

così richiamati dalle classi della business logic

* Il vantaggio relativo all'uso del DAO è dunque il

mantenimento di una rigida separazione tra le

componenti di un'applicazione, le quali potrebbero

essere il "Modello" e il "Controllo" in un'applicazione

basata sul paradigma del MVC.

Si utilizza una fabbrica DAO in modo che sia possibile accedere condizionatamente ai dati da più origini, come un file Proprietà o il database MySQL,

promuovendo così i principi del polimorfismo.

1. Value Object (VO) Crea un oggetto che è un attributo di un altro oggetto senza alcuna identità relativa al dominio e che può essere intercambiabile tra oggetti che usano lo stesso

attributo.

Gli Value Object sono oggetti piccoli, altamente focalizzati e immutabili. Sono sempre di tipo valore. Se è necessario confrontare due VO (per l'uguaglianza), il contenuto deciderà la sua uguaglianza. VO possono migliorare il design, la riusabilità e il codice qualità. Un VO è "usato" per incapsulare i dati aziendali. Nella libreria , i VO vengono utilizzati dagli oggetti business, DAO e JSP.

1. SOFTWARE/OBJECT DESIGN  
   1. UML class diagrams

